Beispieldokumentation

Deutsche Beschreibung

NUTZUNGSBEDINGUNGEN

Die Verwendung der Beispielprogramme erfolgt ausschließlich unter Anerkennung folgender Bedingungen durch den Benutzer:

INSEVIS bietet kostenlose Beispielprogramme für die optimale Nutzung der S7-Programmierung und zur Zeitersparnis bei der Programmerstellung. Für direkte, indirekte oder Folgeschäden des Gebrauchs dieser Software schließt INSEVIS jegliche Gewährleistung genauso aus, wie die Haftung für alle Schäden, die aus die aus der Weitergabe der die Beispielinformationen beinhaltenden Software resultieren.

BEISPIELBESCHREIBUNG DIO8Z

Konfigurationen und Setups

Die PM-DIO8Z ist derzeit in 3 Konfigurationen betreibbar:
- als Vor- und Rückwärtszähler
- zur Frequenz- und Zeitmessung
- als Synchron Serielles Interface (SSI).

Im Hostgerät kann die Konfigurationen über das Tool ConfigStage ausgewählt und geladen werden, als Dezentrale Peripherie ist die Konfiguration "Zähler" Standard, kann auf Anfrage oder in einem Hostgerät aber umprogrammiert werden.

Befindet sich das PM im Hostgerät, kann für alle Konfigurationen über das Tool ConfigStage ein Setup eingestellt werden. Dieses Setup ist jedoch auch zur Laufzeit unter S7 programmierbar. Dabei hilfet folgendes Programmbeispiel.

S7-Programm für Setups

Um das Programmieren des Setups zu vereinfachen, beinhaltet das Beispiel verschiedene Bausteine, je nach Konfiguration und ob im Hostgerät (zentral) oder dezentral.

FC100 Setup Frequenzmessung (zentral = onboard):

```
CALL  "Setup_DIO8Z_FTM"                FC100
 base       :="base_dio8Z_FTM"         Startadresse DIO8Z
 index      :=B#16#0                   0: Zähler 0, 1: Zähler 1
 DO_ena     :="DO_Ena_FTM"             enable globale DO-Bits
 DO_val     :="DO_val_FTM"             Wert globale DO-Bits
 compare    :="compare_FTM0"           Vergleichswert
 itime      :="itime_FMT0"             ITime (Messzeit für Frequenzmessung)
 gate       :="swgate_FTM0"            Software-Gate (1 = Freigabe)
 mode_INK   :="mode_INK_FTM0"          0: Puls/Dir 1: Inkrementalgeber
 mode_T     :="mode_T_FMT0"            0: Frequenz-, 1: Zeitmessung (Periodendauer)
 mode_Tstep:="mode_Step_FTM0"          Zeitbasis Periodendauer 0: 1ms 1: 40ns
 FLim       :="FLIM_FTM0"              Eingangsfilter 0: 1µs 1: 16µs, 2: 64µs, 3: 256µs
```

FC101 Setup Frequenzmessung (dezentral über CAN-DP)

```
CALL  "Setup_DIO8Z_FTM_CAN"            FC101
 base       :="base_dio8Z_FTM"         Startadresse DIO8Z
 index      :=B#16#0                   0: Zähler 0, 1: Zähler 1
 DO_ena     :="DO_Ena_FTM"             enable globale DO-Bits
 DO_val     :="DO_val_FTM"             Wert globale DO-Bits
 compare    :="compare_FTM0"           Vergleichswert
 itime      :="itime_FMT0"             ITime (Messzeit für Frequenzmessung)
 gate       :="swgate_FTM0"            Software-Gate (1 = Freigabe)
 mode_INK   :="mode_INK_FTM0"          0: Puls/Dir 1: Inkrementalgeber
 mode_T     :="mode_T_FMT0"            0: Frequenz-, 1: Zeitmessung (Periodendauer)
```

```
mode_Tstep:="mode_Step_FTM0"   Zeitbasis Periodendauer 0: 1ms 1: 40ns
FLim       :="FLIM_FTM0"       Eingangsfilter 0: 1µs 1: 16µs, 2: 64µs, 3: 256µs
state      :="InitState2"      interner Zustandsmerker;
                               0: inaktiv, fertig
                               1: Start (auf 1 setzen zum Aktivieren)
                               >1: aktiv
```

FC200 Setup Zähler (zentral = onboard)

```
CALL  "Setup_DIO8Z CNT"        FC200
 base    :="base_dio8Z_CNT"    Startadresse DIO8Z
 index   :=B#16#0              0: Zähler 0, 1: Zähler 1
 DO_ena  :="DO_Ena_CNT"        enable globale DO-Bits
 DO_val  :="DO_val_CNT"        Wert globale DO-Bits
 preload :="preload_CNT0"      Zähler Initialisierungs (Start-)wert
 compare :="compare_CNT0"      Vergleichswert
 mode_INK:="mode_INK_CNT0"     0: Puls/Dir 1: Inkrementalgeber
 gate    :="swgate_CNT0"       Software-Gate (1 = Freigabe)
 FLim    :="FLIM_CNT0"         Eingangsfilter 0: 1µs 1: 16µs, 2: 64µs, 3: 256µs
```

FC201 Setup Zähler (dezentral über CAN-DP)

```
CALL  "Setup_DIO8Z CNT CAN"    FC201
 base    :="base_dio8Z_CNT"    Startadresse DIO8Z
 index   :=B#16#0              0: Zähler 0, 1: Zähler 1
 DO_ena  :="DO_Ena_CNT"        enable globale DO-Bits
 DO_val  :="DO_val_CNT"        Wert globale DO-Bits
 preload :="preload_CNT0"      Zähler Initialisierungs (Start-)wert
 compare :="compare_CNT0"      Vergleichswert
 mode_INK:="mode_INK_CNT0"     0: Puls/Dir 1: Inkrementalgeber
 gate    :="swgate_CNT0"       Software-Gate (1 = Freigabe)
 FLim    :="FLIM_CNT0"         Eingangsfilter 0: 1µs 1: 16µs, 2: 64µs, 3: 256µs
 state   :="InitState0"        interner Zustandsmerker;
                               0: inaktiv, fertig
                               1: Start (auf 1 setzen zum Aktivieren)
                               >1: aktiv
```

FC300 Setup SSI (zentral = onboard)

```
CALL  "Setup_DIO8Z SSI"        FC300
 base       :="base_dio8Z_SSI" Startadresse DIO8Z
 index      :=B#16#0           0: Interface 0, 1: Interface 1
 DO_ena     :="DO_Ena"         enable globale DO-Bits
 DO_val     :="DO_val"         Wert globale DO-Bits
 number_Bits:="noBits_SSI0"    Anzahl Taktimpulse (1 … 32)
 Graycode   :="Gray_SSI0"      0: Dual Code, 1: Gray Code
 Pause      :="pause_SSI0"     Taktbüschelpause 0: 64 µs 1: 32 µs, 2: 16 µs, 3: 8 µs
 Latch      :="Latch_SSI0"     Latch: 0: disabled 1: high, 2: low 3: edge
 ClkFrqu    :="ClkFrq_SSI0"    Taktfrequenz 0: disabled 1: 62,5 kHz, 2: 125 kHz
                               3: 250 kHz 4: 500 kHz, 5: 1 MHz, 6: 1,5 MHz, 7: 2MHz
```

FC301 Setup SSI (dezentral über CAN-DP)

```
CALL  "Setup_DIO8Z SSI CAN"    FC300
 base       :="base_dio8Z_SSI" Startadresse DIO8Z
 index      :=B#16#0           0: Interface 0, 1: Interface 1
 DO_ena     :="DO_Ena"         enable globale DO-Bits
 DO_val     :="DO_val"         Wert globale DO-Bits
 number_Bits:="noBits_SSI0"    Anzahl Taktimpulse (1 … 32)
 Graycode   :="Gray_SSI0"      0: Dual Code, 1: Gray Code
 Pause      :="pause_SSI0"     Taktbüschelpause 0: 64 µs 1: 32 µs, 2: 16 µs, 3: 8 µs
 Latch      :="Latch_SSI0"     Latch: 0: disabled 1: high, 2: low 3: edge
```

ClkFrqu	:="ClkFrq_SSI0"	Taktfrequenz 0: disabled 1: 62,5 kHz, 2: 125 kHz 3: 250 kHz 4: 500 kHz, 5: 1 MHz, 6: 1,5 MHz, 7: 2MHz
state	:="InitState4"	interner Zustandsmerker; 0: inaktiv, fertig 1: Start (auf 1 setzen zum Aktivieren) >1: aktiv

Zählertest für DIO8Z-24V

Um die Zählerfunktion und die Konfiguration zu testen, beinhaltet das Beispiel eine Testfunktion.
Dafür ist es notwendig dass die DIO8Z in der 24V-Variante vorliegt und im System eine DIO16verfügbar ist und dass diese mit 4 Pins mit der DIO8Z folgend verdrahtet ist

DIO16 0.1 (Pin 3) ↔ DIO8Z DIO0+ (Pin 3)
DIO16 0.3 (Pin 5) ↔ DIO8Z DIO1+ (Pin 5)
DIO16 1.1 (Pin 13) ↔ DIO8Z DIO4+ (Pin 13)
DIO16 1.3 (Pin 15) ↔ DIO8Z DIO5+ (Pin 15)

(24V-Versorgung stillschweigend vorausgesetzt)
Die Konfiguration mit der ConfigStage muss dem Hardwareausbau entsprechend erfolgen, die Vergabe der Peripherieadressen ist frei. Die Parametrierung des Weckalarms OB35 bestimmt den Takt des Tests, für eine DIO8Z als Dezentrale Peripherie muss dieser mindestens auf 20ms eingestellt sein.
Die Startadressen der DIO16 sowie der DIO8Z müssen später in den Test-Variablentabellen eingetragen werden.

Im OB35 werden je nach Freigabebits Bitmuster für 2 unabhängige Zähler vor und rückwärts erzeugt.
Im OB1 erfolgt nur der Aufruf der Setup-Bausteine.
In den vorbereiteten Variablentabellen wird das setup eingestellt und gestartet bzw werden die Bitmuster zum Zählen freigegeben. Die Kontrolle der Zähler erfolgt manuell. (ED an konfigurierter Peripherieadresse bzw. +4)

Utility

Als weitere Hilfmittel gibt es
FC400 Clear NDR-Flags (Frequenzmessung)
und
FC500 ReadPeripodic (s.u.)

FC500 Periodisch Zählen

Gelegentlich soll die gezählte Position als periodischer Winkel 0..360° oder 180° ausgegeben werden.
Der Geber hat aber eine andere Strichzahl (z.B: 1000 Pulse / U), die in 360° umgerechnet werden kann.

Bei der notwendigen Modulo-Berechung entsteht ein Problem:
Nach vielen Umdrehungen läuft der 32Bit-Zähler über und da die Strichzahl i.d.R. keine 2er-Potenz ist, gibt es keine allgemeine Formel zum umrechnen.

Softwarelösung:
Wir merken uns als Offset den Zählerwert der letzten vollen Umdrehung (0°).
Wenn der neue Zählerwert die Periodengrenzen überschreitet, wird der Offset nachgezogen
d.h. die Startposition der letzten Umdrehung um +/- 1 Periode korrigiert.
Falls der FC nicht mindestens 1x pro Umdrehung aufgerufen werden konnte, muss eine Schleife programmiert werden. (Zykluszeit !!)

Bedingung:
Der FC muss mindestens 1x pro 31 Bit Inkrementen aufgerufen werden (Das sollte machbar sein).

Pseudo-Code:

```
OUT = Counter (PED) - OFFSET
while( OUT > PERIODE_MAX)
        OFFSET += (PERIODE_MAX - PERIODE_MIN)
        OUT = Counter (PED) - OFFSET
```

```
        while( OUT < PERIODE_MIN)
                OFFSET -= (PERIODE_MAX - PERIODE_MIN)
                OUT = Counter (PED) - OFFSET
```

FC500 Setup Periodic count (zentral = onboard))

`CALL "RdPeriodic"`		FC500
`base`	`:="base_dio8Z_SSI"`	Startadresse DIO8Z
`index`	`:=B#16#0`	0 für Zähler 0, 1 für Zähler 1
`RESET`	`:="PeriodOffsReset"`	Reset Periode
`Per_Max`	`:="PeriodMax"`	Maximalwert der Periode
`Per_Min`	`:="PeriodMin"`	Minimalwert der Periode
`Out`	`:="PeriodOut"`	Rückgabewert
`Offset`	`:="PeriodOffs"`	Offset der Periode

PS: Der Zähler-Überlauf funktioniert nur, weil auch die S7-Register bei 32 Bit überlaufen

Hinweis:
- noch keine -

RÜCKMELDUNGEN
Möchten Sie Erweiterungswünsche oder Fehler zu diesen Beispielen melden oder wollen Sie anderen eigene Bespielprogramme kostenlos zur Verfügung stellen? ***Bitte informieren Sie uns unter info@insevis.de***
Gern werden Ihre Programme -auf Wunsch mit Benennung des Autors- allen INSEVIS- Kunden zur Verfügung gestellt.

English description

TERMS OF USE

The use of this sample programs is allowed only under acceptance of following conditions by the user:
The present software which is for guidance only aims at providing customers with sampling information regarding their S7-programs in order to save time. As a result, INSEVIS shall not be held liable for any direct, indirect or consequential damages respect to any claims arising from the content of such software and/or the use made by customers of this sampling information contained herin in connection with their own programs.

SAMPLE DESCRIPTION DIO8Z

Configurations and Setups
The PM-DIO8Z is currently available in 3 configurations:
- as up- and downcounter,
- for frequency- and time measurement and
- as Synchron Serial Interface (SSI)

The configuration can be choosed in the tool ConfigStage and downloaded when the DIO8Z is placed as central periphery. Using DIO8Z as decentral periphery configuration up-and-downcounter will be default.

The Tool ConfigStage includes for all configurations a setup dialog. These setup can be programmed with S7 as well.

S7-program for setup
To ease the programming the setups these example contains some fuction blocks - depending your configuration and depending central or decentral.

FC100 Setup frequency and time measurement (onboard)
FC101 Setup frequency and time measurement (dezentral over CAN-DP)
FC200 Setup Up-Down-Counter (onboard)
FC201 Setup Up-Down-Counter (dezentral over CAN-DP)

Countertest
To validate counting and setup the example contains a system-selftest function. This requires a DIO16 in the PLC system, wired with 4 Pins to DIO8Z (proper 24V-supply assumed):

DIO16 0.1 (Pin 3) ↔ DIO8Z DIO0+ (Pin 3)
DIO16 0.3 (Pin 5) ↔ DIO8Z DIO1+ (Pin 5)
DIO16 1.1 (Pin 13) ↔ DIO8Z DIO4+ (Pin 13)
DIO16 1.3 (Pin 15) ↔ DIO8Z DIO5+ (Pin 15)

The configuration data in ConfigStage must be according your hardware assembly, the specification of used I/O addresses is free. The CPU-parameter of cyclic interrupt OB35 determines the clock of the tests.
In case of a DIO8Z as decentrale periphery it must be at least 20ms.

The I/O addresses of DIO16 and DIO8Z must be typed in later into variable tables.

Cyclic interrupt OB35 produces output patterns for 2 counters regarding counter modes back and forward.
OB1 just calls the setup-functions.
In the prepared Vartabs you will find predefined setups to load and control flags to run the patterns. To verify counter results manually adjust tzhe I/O addresses and watch both ED's

Utility
There are also
FC400 Clear NDR-Flags (for frequeny measure)
and
FC500 ReadPeripodic (s.below.)

FC500 Count Periodicly

Sometimes it is usefull to get a position as cyclical angle 0..360° or 180°.
Maybe the encode never meets a power-by-2 pulses per round to allow binary masking.
A modulo operation crashes in case of the counters overflow (turn around 32 bits).

Solution:
Store an offset with the position of the last round. (0°).
If the counter ecxceeds the periods limit, increase / decrease the offset of one round and always return the difference between counter and offset
In case the FC is not colled once per round place a loop inside (Caution with bad- or ininitialized values !)
Remaining Condition : The FC must be called once in a 32bit cycle

Pseudo-Code:

```
OUT = Counter (PED) - OFFSET
while( OUT > PERIODE_MAX)
        OFFSET += (PERIODE_MAX - PERIODE_MIN)
        OUT = Counter (PED) - OFFSET

while( OUT < PERIODE_MIN)
        OFFSET -= (PERIODE_MAX - PERIODE_MIN)
        OUT = Counter (PED) - OFFSET
```

Hint:
-

FEEDBACK
Do you want to inform us about necessary increments or errors or do you want to provide us with your sample programs to offer it for free to all customers? ***Please inform us at info@insevis.de***
Gladly we would provide your program -if you wish with the authors name- to all other customers of INSEVIS.